Interpolation-restart strategies for resilient eigensolvers

Emmanuel Agullo, Luc Giraud, Pablo Salas, Mawussi Zounon

INTERPOLATION-RESTART STRATEGIES FOR RESILIENT EIGENSOLVERS

E. AGULLO*, L. GIRAUD*, P. SALAS†, AND M. ZOUNON*‡

Abstract. The solution of large eigenproblems is involved in many scientific and engineering applications when for instance, stability analysis is a concern. For large simulation in material physics or thermo-acoustics, the calculation can last for many hours on large parallel platforms. On future large-scale systems, the mean time between failures (MTBF) of the system is expected to decrease so that many faults could occur during the solution of large eigenproblems. Consequently, it becomes critical to design parallel eigensolvers that can survive faults. In that framework, we investigate the relevance of approaches relying on numerical techniques, which might be combined with more classical techniques for real large-scale parallel implementations. Because we focus on numerical remedies we do not consider parallel implementations nor parallel experiments but only numerical experiments. We assume that a separate mechanism ensures the fault detection and that a system layer provides support for setting back the environment (processes, ...) in a running state. Once the system is in a running state, after a fault, our main objective is to provide robust resilient schemes so that the eigensolver may keep converging in the presence of the fault without restarting the calculation from scratch. For this purpose, we extend the interpolation-restart (IR) strategies initially introduced for the solution of linear systems in a previous work to the solution of eigenproblems in this paper. For a given numerical scheme, the IR strategies consist of extracting relevant spectral information from available data after a fault. After data extraction, a well-selected part of the missing data is regenerated through interpolation strategies to constitute a meaningful input to restart the numerical algorithm. One of the main features of this numerical remedy is that it does not require extra resources, *i.e.*, computational unit or computing time, when no fault occurs. In this paper, we revisit a few state-of-the-art methods for solving large sparse eigenvalue problems namely the Arnoldi methods, subspace iteration methods and the Jacobi-Davidson method, in the light of our IR strategies. For each considered eigensolver, we adapt the IR strategies to regenerate as much spectral information as possible. Through extensive numerical experiments, we study the respective robustness of the resulting resilient schemes with respect to the MTBF and to the amount of data loss via qualitative and quantitative illustrations.

keyword numerical resiliency; hard fault; fault tolerance; eigenvalue problems; linear algebra; HPC; numerical methods; Arnoldi; IRAM; subspace iteration; Jacobi-Davidson.

1. Introduction. The computation of eigenpairs (eigenvalues and eigenvectors) of large sparse matrices is involved in many scientific and engineering applications such as when stability analysis is a concern. To name a few, it appears in structural dynamics, thermodynamics, thermo-acoustics, quantum chemistry. With the permanent increase of the computational power of high performance computing (HPC) systems by using a larger and larger number of CPU cores or specialized processing units, HPC applications are increasingly prone to faults.

To guarantee fault tolerance, two classes of strategies are required. One for the fault detection and the other for fault correction. Faults such as computational node crashes are obvious to detect while silent faults may be challenging to detect. To cope with silent faults, a duplication strategy is commonly used for fault detection [18, 39] by comparing the outputs, while triple modular redundancy (TMR) is used for fault detection and correction [34, 37]. However, the additional computational resources required by such replication strategies may represent a severe penalty. Instead of replicating computational resources, studies [7, 36] propose a time redundancy model for fault detection. It consists in repeating computation twice on the same resource. The advantage of time redundancy models is the flexibility at application level; software developers can indeed select only a set of critical instructions to protect. Recomputing only some instructions instead of the whole application lowers the time redundancy overhead [25]. In some numerical simulations, data naturally satisfy well defined mathematical properties. These properties can be efficiently exploited for fault detection through a periodical check of the numerical properties during computation [10].

Checkpoint/restart is the most studied fault recovery strategy in the context of HPC systems. The common checkpoint/restart scheme consists in periodically saving data onto a reliable storage device such as a remote disk. When a fault occurs, a rollback is performed to the point of the most recent and consistent checkpoint. According to the implemented checkpoint strategy, all processes

*Inria Bordeaux Sud-Ouest, 200 avenue de la Vieille Tour, F-33400 Talence, France
†Sherbrooke University, Canada
‡Current affiliation: The University of Manchester, UK

may perform the periodical record simultaneously. It is called a coordinated checkpoint [12, 27, 33]. In parallel distributed environments, synchronizations due to coordination may significantly degrade application performance [13, 24]. To avoid synchronization, an uncoordinated checkpoint may be employed combined with message logging protocols [4, 8, 20]. Many mechanisms have been developed to lower the overhead of the checkpoint/restart strategy [23, 28, 38]. However, the additional usage of resources (such as memory, disk) that is required by checkpoint/restart schemes may be prohibitive, or the time to restore data might become larger than the MTBF [9].

Algorithm-based fault tolerance (ABFT) is a class of approaches in which algorithms are adapted to encode extra data for fault tolerance at expected low cost [5, 15, 17, 26]. The basic idea consists in maintaining consistency between extra encoded data and application data. The extra encoded data can be exploited for fault detection and for recovery of lost data. ABFT strategies may be excellent candidates to ensure the resilience of an application; however they induce extra costs for computing and storing the data encoding even when no fault occurs.

To cope with unstable situations, numerical simulations might also to be equipped with numerical resilient mechanisms that enable them to complete a calculation even when several faults occur. In this paper, we present a new class of numerical fault tolerance algorithms in the framework of the solution of eigenproblems. This new resilient scheme is designed at application level and does not require extra resources, i.e., computational unit or computing time, when no fault occurs. We consider the standard eigenproblem of the form:

$$Au = \lambda u,$$

where $A \in \mathbb{C}^{n\times n}$, with $u \neq 0$, $u \in \mathbb{C}^n$ and $\lambda \in \mathbb{C}$. The couple (λ, u) is called an eigenpair of A where the vector u is an eigenvector with the associated eigenvalue λ.

In this paper, we extend the interpolation-restart (IR) strategies introduced for the solution of linear systems [1, 21, 3] to a few state-of-the-art eigensolvers. More precisely, the Arnoldi [6], Implicitly restarted Arnoldi [22], subspace iteration [30], and Jacobi-Davidson [35] algorithms are being revisited to make them resilient in the presence of faults. Most of the considered eigensolvers naturally implement a restart mechanism to cope with memory constraints. We incorporate an additional restart in our scheme after the faults. We attempt to regenerate as much relevant spectral information as possible to perform the restart effectively and efficiently.

The remainder of the paper is structured as follows. In Section 2 we describe how the interpolation techniques can be applied to regenerate meaningful spectral information. We briefly present the eigensolvers that we have considered in Section 3 and detail how the recovery ideas can be tuned for each one of them. Section 4 is devoted to the numerical experiments; we discuss the robustness of the various resilient numerical schemes and conclude with some perspectives in Section 5.

2. Interpolation-restart principles. In this section, we describe how the interpolation strategies can be used to regenerate meaningful spectral information. Contrarily to what has been proposed for the Krylov linear solvers where only a meaningful iterate is computed to serve as a new initial guess for restarting the iterations [1, 21, 41], more flexibility exists in the framework of eigensolution where similar ideas can be adapted to better exploit the numerical features of the individual eigensolvers. The main reasons are that some of the considered eigensolvers do not rely on a central equality or a sophisticated short-term recurrence (such as Conjugate Gradient for linear system solution). In particular, when a few eigenpairs are computed some additional opportunities exists to recompute meaningful spectral information. We present in details different variants for selecting and computing the relevant subspaces to perform the restart for each particular considered eigensolver in Section 3.

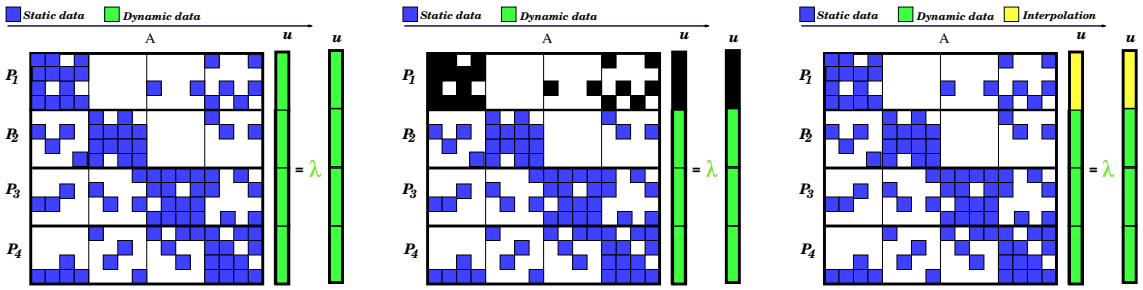
2.1. Context. We propose to design and study resilient parallel eigensolvers. We assume that a separate mechanism (orthogonal to this study) ensures fault detection and we focus on the design of IR strategies. For the sake of exposure, we furthermore consider a parallel distributed

memory context to present the proposed IR (although these strategies can be extended to other HPC contexts [19]).

ASSUMPTION 1. *In our parallel computational context, all the vectors or matrices of dimension n are distributed by blocks of rows in the memory of the different computing nodes but scalars or low dimensional matrices are replicated.*

According to Assumption 1, the eigenvector u and the matrix A are distributed according to a block row partition as well as all vectors of dimension n generated during the solution whereas scalars (for example λ) or low dimensional matrices are replicated on all nodes. Let N be the number of partitions, such that each block row is mapped to a computing node. For all p, $p \in [1, N]$, I_p denotes the set of row indices mapped to node p. With respect to this notation, node p stores the block row $A_{I_p,:}$ and u_{I_p} as well as the entries of all the vectors involved in the solver associated with the corresponding row indices of this block row. If the block A_{I_p,I_q} contains at least one non zero entry, node p is referred to as neighbor of node q as communication will occur between those two nodes to perform a parallel sparse matrix-vector product.

When a fault occurs on a node, all data in its memory are lost. We consider the formalism proposed in [21] where lost data are classified into three categories: the *computational environment*, the *static* data, and the *dynamic* data. The computational environment is all the data needed to perform the computation (code of the program, environment variables, ...). The static data are those that are set up during the initialization phase and that remain unchanged during the computation such as the coefficient matrix A. The dynamic data is all data whose value may change during the computation. The iterates λ and u are examples of dynamic data. In Figure 2.1a, we depict a block row distribution on four nodes. The data in blue is the static data associated with the eigenproblem (*i.e.*, the matrix) while the data in green is the dynamic data (here, only the eigenpair (λ, u) is shown). If node P_1 fails, the first block row of A as well as the first entries of u are lost (in black in Figure 2.1b). The eigenvalue λ is a scalar replicated on all nodes and thus remains available on non-failed ones.



(a) Before fault (b) Faulty iteration (c) Interpolation

Fig. 2.1: General interpolation scheme. The matrix is initially distributed with a block row partition, here on four nodes (a). When a fault occurs on node P_1, the corresponding data is lost (b). Whereas static data can be immediately restored, dynamic data that has been lost cannot and we investigate numerical strategies for interpolating it (c).

We assume that, when a fault occurs, the failed node is replaced and the associated computational environment and static data are restored. In Figure 2.1c for instance, the first matrix block row is restored as it is a static data. However, the eigenvector iterate u, being a dynamic data, its entries u_{I_1} are definitely lost and we present strategies for regenerating them through some interpolations that follow. Our strategies do not attempt to interpolate all the dynamic data but for each eigensolver, we study which meaningful part of the dynamic data could be interpolated. Because not all data is regenerated and the interpolation computed an approximation of the lost

data, part of the information remains lost. As a consequence, it is furthermore required to restart the numerical process. We will present in detail propositions for selecting (or producing) relevant data for performing that restart for each particular eigensolver considered in Section 3. We assume in the rest of this paper that a fault occurs during iteration $f+1$ and the proposed interpolation strategies are thus based on the numerical values computed by the algorithms at iteration f.

2.2. Interpolation methods. The IR strategies consist in interpolating lost data by using non-corrupted data. Let $u^{(f)}$ be an approximated eigenvector when a fault occurs. After the fault, the entries of $u^{(f)}$ are correct, except those stored in the failed node p. Assuming that in a parallel distributed environment, the current eigenvalue λ_f is naturally replicated in the memory of the different computing nodes, we present two strategies to compute a new approximate eigenvector. The first strategy, referred to as linear interpolation and denoted LI, consists in solving a local linear system associated with the submatrices A_{I_p,I_p} of the failed node. The second one relies on the solution of a least squares interpolation and is denoted LSI. Those two alternatives result from considering $(\lambda_f, u^{(f)})$ as an exact eigenpair. We may have a block row viewpoint, which defines the LI variant. If node p fails, LI computes a new approximation of the eigenvector $u^{(LI)}$ as follows

$$\begin{cases} u^{(LI)}_{I_q} = u^{(f)}_{I_q} & \text{for } q \neq p, \\ u^{(LI)}_{I_p} = (A_{I_p,I_p} - \lambda \mathcal{I}_{I_p,I_p})^{-1}(-\sum_{q \neq q} A_{I_p,I_q} u^{(f)}_{I_q}). \end{cases}$$

Alternatively, we can have a block column point of view, which leads to the LSI variant that computes $u^{(LSI)}$ via

$$\begin{cases} u^{(LSI)}_{I_q} = u^{(f)}_{I_q} & \text{for } q \neq p, \\ u^{(LSI)}_{I_p} = \underset{u_{I_p}}{\operatorname{argmin}} \| (A_{:,I_p} - \lambda \mathcal{I}_{:,I_p}) u_{I_p} - \sum_{q \neq p} (A_{:,I_q} - \lambda \mathcal{I}_{:,I_q}) u^{(f)}_{I_q}) \|. \end{cases}$$

Here, $\mathcal{I} \in \mathbb{C}^{n \times n}$ is the identity matrix and we furthermore assume that $(A_{I_p,I_p} - \lambda I_{I_p,I_p})$ is non singular and that $(A_{:,I_p} - \lambda \mathcal{I}_{:,I_p})$ has full rank. The matrix involved in the least squares problem, $(A_{:,I_p} - \lambda \mathcal{I}_{:,I_p})$, is sparse of dimensions $|J_p| \times |I_p|$ where its number of rows $|J_p|$ depends on the sparsity structure of $A_{:,I_p}$. Consequently the LSI strategy may have a higher computational cost. In the rest of this paper, we use the generic acronym IR to refer to any of the interpolation strategies (LI and LSI) where there is no need to make a clear distinction between them.

3. Interpolation-restart strategies for general purpose eigensolvers. In this section, we briefly describe a few of the most popular eigensolvers for general sparse matrices that we have considered in this work to compute a few eigenpairs and denote *nev* this number of eigenpairs. For each eigensolver, we describe which data are regenerated after a fault to make them resilient. We also briefly present the numerical approach and associated algorithm. We then describe how the IR strategies can be applied to compute a few of the key numerical quantities of the solver when some data is lost as well as how this data can be used to perform an effective and efficient restart.

3.1. Subspace iterations to compute a few eigenpairs.

Brief description of the numerical algorithm. The subspace iteration method is a block variant of the power method [16]. To compute a few (*nev*) eigenpairs it starts with an initial block of $nev+s$ (with $s \geq 0$) linearly independent vectors corresponding to matrix $U^{(0)} = [u^{(0)}_1, \ldots, u^{(0)}_{nev+s}] \in \mathbb{C}^{n \times (nev+s)}$. Under certain assumptions [30], the sequence of $U^{(k)} = A^k U^{(0)}$ generated by the algorithm, converges to the $nev + s$ eigenpairs of A associated with the eigenvalues of the largest magnitude. To guarantee the full column rank in $U^{(k)}$ for large values of k, the Q factor of its QR factorization may be used at each iteration.

Algorithm 1 Basic subspace iteration

1: Choose $U^{(0)} = [u_1^{(0)}, ..., u_{nev+s}^{(0)}] \in \mathbb{C}^{n\times(nev+s)}$
2: **for** $k = 0 \dots$ until convergence **do**
3: Orthonormalize $U^{(k)}$
4: Compute $W^{(k)} = AU^{(k)}$
5: Form the Rayleigh quotient $C^{(k)} = {W^{(k)}}^H AW^{(k)}$
6: Compute the eigenvectors $G^{(k)} = [g_1, \dots, g_{nev+s}]$ of $C^{(k)}$ and eigenvalues $\sigma(C^{(k)}) = (\lambda_1, \dots, \lambda_{nev+s})$
7: Update Ritz vectors : $U^{(k+1)} = W^{(k)}G^{(k)}$
8: **end for**

Interpolation-restart policy. In the subspace iteration method depicted in Algorithm 1, according to Assumption 1, the Ritz vectors $U^{(k)}$ are distributed, whereas the Rayleigh quotient $C^{(k)}$ and Ritz values are replicated. When a fault occurs, we distinguish two cases. During an iteration, a fault may occur before or after the computation of the Rayleigh quotient $C^{(f+1)}$.

1. When a fault occurs before the computation of the Rayleigh quotient $C^{(f+1)}$ (Algorithm 1, lines 2 to 5) the surviving nodes cannot compute the Rayleigh quotient $C^{(f+1)}$ because entries of $W^{(f+1)}$ are missing. In this case, we consider the available entries of the Ritz vectors $U^{(f)}$ and its corresponding eigenvalues $\sigma(C^{(f)})$. We interpolate the m Ritz vectors individually $(u_\ell^{(f)}, 1 \leq \ell \leq nev + s)$ using LI or LSI.
2. When a fault occurs after the computation of the Rayleigh quotient $C^{(f+1)}$ (Algorithm 1, line 6 to 7) all surviving nodes can compute the entries of $U^{(f+1)}$ relying on a local replicate of $C^{(f+1)}$ and the local entries of $W^{(f+1)}$. The missing entries of each Ritz vector $(u_\ell^{(f+1)}, 1 \leq \ell \leq nev + s)$ can be individually interpolated using LI or LSI relying on the corresponding eigenvalues $\sigma(C^{(f+1)})$.

After the interpolation, the subspace iteration algorithm is restarted with the matrix $U^{(IR)} = [u_1^{(IR)}, \dots, u_{nev+s}^{(IR)}] \in \mathbb{C}^{n\times(nev+s)}$ until convergence or the next fault.

3.2. Arnoldi method to compute one eigenpair.

Brief description of the numerical algorithm. The Arnoldi method is an efficient procedure for approximating an eigenvalue lying in the periphery of the spectrum of A. For a prescribed dimension m, referred to as the restart, the method starts from an initial guess vector v_1 of norm one and builds an orthonormal basis V_m of the search space spanned by $span\{v_1, Av_1, ..., A^{m-1}v_1\}$ as well as an upper Hessenberg matrix $H_m \in \mathbb{C}^{m\times m}$. Those matrices satisfy the so-called Arnoldi equality:

$$AV_m = V_m H_m + \beta_m v_{m+1} e_m^T, \tag{3.1}$$

where e_m denotes the last column of the $m \times m$ identity matrix. By construction, H_m is the Rayleigh quotient matrix associated with V_m that is used to compute approximate eigenpairs of A. If the computed Ritz pair corresponding to the targeted eigenvalue is not accurate enough, the procedure is restarted using the best current Ritz vector as the new initial guess. This space expansion and restart sequence is repeated until convergence.

Interpolation-restart policy. According to Assumption 1, we assume that the low dimension Hessenberg matrix H_k is replicated on each node. Consequently, regardless of the step in which the fault occurs during the iteration, each surviving node q can solve the eigenproblem $H_f g = \lambda g$ redundantly, then compute its entries of the Ritz vector $u_{I_q} = V_f(I_p, :)g$. The next step is the computation of the Ritz vector entries allocated on the failed node using LI or LSI. The resulting vector becomes the new initial guess to restart the Arnoldi iterations.

3.3. Implicitly restarted Arnoldi method to compute a few eigenpairs.

Brief description of the numerical algorithm. Developed by Lehoucq and Sorensen in [22], the implicitly restarted Arnoldi method (IRAM) depicted in Algorithm 2 is commonly used for the solution of large eigenvalue problems. IRAM is an extension of the Arnoldi method that starts with an Arnoldi equality of size m. From the Arnoldi equality of dimension m, IRAM performs a contraction of the equality from size m down to size $\tilde{m}$ ($nev \leq \tilde{m} \leq m$). This is achieved by applying a polynomial filter of degree $\ell = m - \tilde{m}$ that reduces the size of the Arnoldi equality down to $\tilde{m}$ (see Algorithm 2, line 12), that is then expanded again to size m before checking the accuracy of the Ritz eigenpairs. The expansion and contraction steps are repeated until convergence. The contraction step acts as a filter to focus the spectral information in a target region of the spectrum from the m dimensional Krylov subspace while maintaining the Arnoldi equality.

Algorithm 2 Implicitly restarted Arnoldi method with restart m

1: Compute Arnoldi equality $AV_m = V_m H_m + f_m e_m^T$.
2: **for** $k = 0, 1, \ldots,$ until convergence, **do**
3: Compute $\sigma(H_m)$ and select ℓ shifts $\mu_1, \ldots, \mu_\ell$ ($\ell = m - \tilde{m}$).
4: $Q = \mathcal{I}_m$
5: **for** $i = 1, \ldots, \ell$ **do**
6: QR Factorize $Q_i R_i = H_m - \mu_i I$
7: $H_m = {Q_i}^H H_m Q_i$
8: $Q = Q Q_i$
9: **end for**
10: $\beta_{\tilde{m}} = H_m(\tilde{m}+1, \tilde{m})$
11: $f_{\tilde{m}} = v_{\tilde{m}+1}\beta_{\tilde{m}} + f_m Q(m, \tilde{m})$
12: $V_{\tilde{m}} = V_m Q(:, 1:\tilde{m})$; $H_{\tilde{m}} = H_m(1:\tilde{m}, 1:\tilde{m})$
13: Starting with $AV_{\tilde{m}} = V_{\tilde{m}} H_{\tilde{m}} + f_{\tilde{m}} e_{\tilde{m}}^T$,
perform ℓ steps of Arnoldi algorithm to get $AV_m = V_m H_m + f_m e_m^T$
14: **end for**

Interpolation-restart policy. When a fault occurs during iteration $f+1$, it may be during the first Arnoldi iteration (Algorithm 2, line 1), the expansion of the Krylov subspace (Algorithm 2, line 13) or during the contraction step (Algorithm 2, lines 3 to 12).

1. When the fault occurs during the first Arnoldi iteration, we simply apply the policy described in Section 3.2.
2. The contraction implicitly implements the shifted QR mechanism. When a fault occurs during the contraction step each surviving node q can compute this step concurrently and redundantly using the replicated Hessenberg matrix H_m. The outcome of this local calculation is that all surviving nodes have $V_{\tilde{m}}(I_q, :)$ as well as the reduced Hessenberg matrix $H_{\tilde{m}}$.
3. When the fault occurs in the expansion phase during step $f+1$ with $\tilde{m} < f+1 < m$, each surviving node can compute the implicitly shift QR update of H_f to compute $H_{\tilde{m}}$ using the shift defined by $\sigma(H_f)$. Using the result of the shift QR calculations, each node q can also compute $V_{\tilde{m}}(I_q, :)$.

From $V_{\tilde{m}}(I_q, :)$ and $H_{\tilde{m}}$, the surviving nodes may then compute eigenvectors $G = [g_1, \ldots, g_{nev}]$ and eigenvalues $(\lambda_1, \ldots, \lambda_{nev})$ of $H_{\tilde{m}}$. Consequently, the entries of the Ritz vectors are computed by

$$U^{(f)}(I_q, :) = V_{\tilde{m}}(I_q, :) G. \tag{3.2}$$

The missing entries I_p of Ritz vectors may be interpolated using either LI or LSI of the interpolation algorithms. In that situation, the available entries of $V_{\tilde{m}}$ no longer satisfy any Arnoldi equality. To take into account all the available spectral information, we compute the linear combination of the interpolated eigenvectors, $u = \sum_{j=1}^{\tilde{m}} u_j^{(IR)}$, and restart with the normalized linear combination

$v_1 = \frac{u}{\|u\|}$ as initial vector. The motivation is that Arnoldi converges within the first expansion if started from a vector v that is a linear combination of $k \leq m$ eigenvectors [16].

3.4. The Jacobi-Davidson method to compute a few eigenpairs.

Brief description of the numerical algorithm. The Jacobi–Davidson method, proposed by Sleijpen and van der Vorst in [35], is a widely used eigensolver. The basic ingredient of Jacobi-Davidson is depicted in Algorithm 3 for the computation of one eigenpair whose eigenvalue is close to a given target τ. It starts with a given normalized vector v and constructs a basis V extended using the Jacobi orthogonal correction method. At each iteration the algorithm computes the Ritz pairs associated with V and selects the eigenpair whose eigenvalue is the closest to the target τ.

Algorithm 3 $[\lambda, u]$ = Basic-Jacobi-Davidson(v, τ)
Jacobi–Davidson algorithm to compute the eigenvalue of A closest to a target value τ

1: Set $V_1 = [v]$
2: **for** $k = 1, 2, \ldots$,until convergence **do**
3: Compute Rayleigh quotient: $C_k = V_k^H A V_k$, and eigenpairs of C_k
4: Select Ritz pair (λ_k, u_k) such that λ_k is the closest to τ
5: $r_k = Au_k - \lambda_k u_k$
6: Perform a few steps of GMRES to solve the correction equation

$$(I - u_k u_k^H)(A - \tau I)(I - u_k u_k^H)v = -r_k, \text{ so that } v \perp u_k$$

7: Compute w by orthonormalizing v against V_k: $w = v - V_k(V_k^H v)$
8: Set $V_{k+1} = [V_k, w]$
9: **end for**

Algorithm 3 converges to one eigenpair. If more than one eigenpair needs to be computed, Algorithm 3 can be accommodated to compute a partial Schur decomposition of A. In that respect, the next iterations are enforced to generate a search space orthogonal to the space spanned by the *nconv* already converged eigenvectors. This is achieved by representing this space using the corresponding Schur vectors. Let $AZ_{nconv} = Z_{nconv}T_{nconv}$ denote the partial Schur form where the columns of the orthonormal matrix Z_{nconv} span the converged eigenspace and the diagonal of the upper triangular matrix T_{nconv} are the associated converged eigenvalues.

Algorithm 4 corresponds to the Jacobi-Davidson style QR algorithm presented in [14]. It is conceived to be used by a higher level routine that decides the number of wanted eigenpairs *nev*, the target point τ, the maximum and the minimum size of the basis V, etc. The inputs for the algorithm are the existing converged Schur vectors Z_{nconv} of A, the current size k of the basis V_k, $W_k = AV_k$, and the Rayleigh quotient C_k. The focal point τ, the maximum dimension m affordable for the space spanned by V, the size of the restarted basis $\tilde{m}$ $(1 \leq \tilde{m} < m)$ and the maximum number of restarts allowed are also provided. Outputs are μ, z and t, such that

$$A(Z_{nconv}\ z) = (Z_{nconv}\ z) \begin{pmatrix} T_{nconv} & t \\ 0 & \mu \end{pmatrix}$$

is a partial Schur decomposition of one higher dimension.

The higher level routine must furnish the necessary inputs to Algorithm 4. If the process starts from the beginning, there are then two situations. The first one corresponds to the case when the computation starts from a single random vector. Then the higher level routine computes an Arnoldi decomposition of size $\tilde{m}$

$$AV_{\tilde{m}} = V_{\tilde{m}} H_{\tilde{m}} + \beta v_{\tilde{m}+1} e_{\tilde{m}}^T,$$

and Jacobi-Davidson starts with $U = [\ ]$, $V = V_{\tilde{m}}$, $W = AV$ and $C = H_{\tilde{m}}$. The second case is when the process starts from a given number k of initial vectors. The initial block of vectors is

Algorithm 4 $[\mu, z, t]$=JDQR(Z(:,1:$nconv$),V(:,1:k),W(:,1:k),C(1:k,1:k), τ,$\tilde{m}$,m,$maxiter$)
Jacobi-Davidson style QR algorithm for expansion of partial Schur decomposition

Set $iter = 0$; $k_{init} = k$; $tr = 0$
while iter<maxiter **do**
  $iter = iter + 1$
  for $k = k_{init}, \ldots, m$ **do**
    % Computation of the Schur decomposition $CQ = QT$
    % so that the eigenvalues on the diagonal of T
    % are sorted by increasing distance to τ
    $[Q, T] = SortSchur(C(1:k, 1:k), \tau, k)$,
    Choose $\mu = T(1,1)$ and $g = Q(:,1)$, the Ritz pair closest to τ
    Approximate eigenvector of A: $z = V(:, 1:k)g$, and Az: $y = W(:, 1:k)g$
    Compute the residual $r = y - \mu z$, orthogonalize it against Z(:,1:$nconv$) and compute its norm: $rnorm$=norm(r)
    % Convergence test:
    if $rnorm$ is small enough **then**
      $nconv = nconv + 1$
      % Prepare outputs and deflate:
      $t = Z^H y$; $V = V(:, 1:k)Q(:, 2:k)$;
      $W = W(:, 1:k)Q(:, 2:k)$; $C = T(2:k, 2:k)$.
      return
    else if $k = m$ **then**
      % Restart:
      $V(:, 1:\tilde{m}) = V(:, 1:m)Q(:, 1:\tilde{m})$;
      $W(:, 1:\tilde{m}) = W(:, 1:m)Q(:, 1:\tilde{m})$;
      $C(1:\tilde{m}, 1:\tilde{m}) = T(1:\tilde{m}, 1:\tilde{m})$;
      $k_{init} = \tilde{m}$
    end if
    %No convergence reached and $k < m$.
    Solve the correction equation:

$$(I - zz^H)(A_\perp - \tau I)(I - zz^H)v = -r$$

    Orthogonalize v against $V(:, 1:k)$ and $Z(:, 1:nconv)$
    % Extend the Rayleigh basis and the Rayleigh quotient:
    $V(:, k+1) = v$, $W(:, k+1) = Av$, $C(k+1,1:k) = v^H W(:,1:k)$,
    $C(1:k, k+1) = V(:,1:k)^H W(:, k+1)$, $C(k+1, k+1) = v^H W(:, k+1)$
  end for
end while

then orthonormalized to obtain V_k and the process can start as indicated previously, with $Z = [\,]$, $V = V_k$, $W = AV_k$ and $C = V_k^H AV_k = V_k^H W$.

Once a partial Schur form of size nev is available, the eigenpairs (λ_ℓ, u_ℓ) (with $\ell = 1, \ldots, nev$) of A can be computed. The eigenvalue λ_ℓ is the Ritz value of T_{nev} associated with the Ritz eigenvector g_ℓ so that $u_\ell = Z_{nev} g_\ell$.

Interpolation-restart policy. According to Assumption 1, the Schur vectors $Z_{nconv} = [z_1, \ldots, z_{nconv}]$, and the basis $V_f = [v_1, \ldots, v_f]$ are distributed among the computing units as the matrix $T_{nconv} \in \mathbb{C}^{nconv \times nconv}$, and the Rayleigh quotient matrix $C_f \in \mathbb{C}^{f \times f}$ are replicated.

The Jacobi-Davidson algorithm enables more possibilities to regenerate a meaningful context for the restart after a fault. There are mainly two reasons. First, the algorithm does not rely on

an equality that is incremented at each iteration such as Arnoldi; preserving such an incremental equality after a fault is very challenging. Second, the algorithm can start from a set of vectors and its convergence will be fast if these vectors are rich in the sought spectral information.

When the fault occurs on node p while $nconv$ ($nconv > 0$) Schur vectors were converged, good approximations of the associated converged eigenvectors can easily be computed as follows. Each non-faulty node q performs:

1. the spectral decomposition of the partial Schur matrix T_{nconv}

$$T_{nconv}G_{nconv} = G_{nconv}D \text{ with } G_{nconv} = [g_1, \ldots, g_{nconv}],$$

2. and the computation of its entries of the converged eigenvectors

$$u_\ell(I_q) = Z_{nconv}(I_p, :)g_\ell \text{ for } \ell = 1, \ldots, nconv.$$

The missing entries of the eigenvectors can be computed using IR to build $U^{(IR)}_{nconv} = [u^{(IR)}_1, \ldots, u^{(IR)}_{nconv}]$.

In addition to $U^{(IR)}_{nconv}$, further information can be extracted from the search space V_f and the Rayleigh quotient matrix C_f. Following the same methodology, spectral information built from C_f and V_f can be computed to generate additional directions to expand the initial search space ($U^{(IR)}_{nconv}$) used to restart the Jacobi-Davidson algorithm. Each non-faulty node q computes:

1. the sorted Schur decomposition of C_f, that writes $C_f\tilde{G}_f = \tilde{G}_f S_f$ so that the eigenvalues on the diagonal of S_f are sorted by increasing distance to τ, (Algorithm 4, line 5),
2. and the entries of the Ritz vectors $\tilde{u}_\ell(I_q) = V_f(I_q, :)\tilde{g}_\ell$ for $\ell = 1, \ldots, s$, where s is the number of Ritz vectors we want to interpolate. Because $\tilde{G}_f$ has been sorted, these vectors may be considered as the s best candidates to expand $U^{(IR)}_{nconv}$. That is, $\tilde{u_1}$ is the Ritz vector associated with $S(1,1)$ which is the Ritz value closest to the target τ, that is improved by Jacobi-Davidson iterations.

In addition, the missing entries I_p of the Ritz vectors $\tilde{u}_\ell$ can be computed using LI or LSI, $U^{(IR)} = [u^{(IR)}_1, \ldots, u^{(IR)}_{nconv}, \tilde{u}^{(IR)}_1, \ldots, \tilde{u}^{(IR)}_s]$. Once $U^{(IR)}$ has been computed, the vectors in $U^{(IR)}$ are then orthonormalized to obtain $V_{restart}$. The Jacobi-Davidson algorithm can be restarted with $Z = [\,]$, $V = V_{restart}$, $W_{restart} = AV_{restart}$, $C = V^H_{restart}W$.

REMARK 1. *Let us assume that the partial Schur decomposition has converged in exact arithmetic ($AZ_{nconv} = Z_{nconv}T_{nconv}$), and that the nconv eigenpairs are exact solutions ($Au_\ell = \lambda_\ell u_\ell$) still in exact arithmetic. Under this assumption, the eigenvectors ($u^{(IR)}_\ell$) computed by IR are the same exact eigenvectors as long as* $\left(A(I_p, I_p) - \lambda_\ell I_{I_p, I_p}\right)$ *is nonsingular or* $\left(A_{:,I_p} - \lambda_\ell \mathcal{I}_{:,I_p}\right)$ *is full column rank for LI and LSI, respectively. As a consequence, if Jacobi-Davidson is restarted with the initial basis V_{nconv} obtained from the orthonormalization of the vectors of $U^{(IR)}_{nconv}$ then, the nconv already converged Schur vectors will be retrieved in the initial basis V_{nconv}. In floating point arithmetic, there is no guarantee of retrieving the already converged nconv Schur vectors by restarting with V_{nconv}, although in practice this is likely to happen as we will see in the numerical experiments.*

Although s has only to satisfy $0 \leq s \leq f$, because of Remark 1, a natural choice for s is $s = nev$ (we interpolate $nconv + nev$ vectors) so that the initial search space after a fault will be at least of dimension nev.

4. Numerical experiments. In this section, we investigate the numerical behavior of the eigensolvers in presence of faults when the IR policies are applied. We define

$$\tilde{\eta}(u^{(\ell)}, \lambda_\ell) = \frac{\|Au^{(\ell)} - \lambda_\ell u^{(\ell)}\|}{|\lambda_\ell|}, \tag{4.1}$$

the scaled residual associated with the approximate eigenpair $(\lambda_\ell, u^{(\ell)})$ for nonzero eigenvalue approximation. This scaled residual is a lower bound of the normwise backward error of the eigenpair $(u^{(\ell)}, \lambda_\ell)$. Given a threshold ε, the widely used stopping criterion to detect convergence is defined by

$$\tilde{\eta}(u^{(\ell)}, \lambda_\ell) \leq \varepsilon.$$

Through a few experiments, we first illustrate the impact of the number of faults and volume of lost data on the convergence history of the solvers for a few selected eigenproblems. More precisely we consider the subspace iteration method, Arnoldi, IRAM and Jacobi-Davidson IR strategies in sections 4.1, 4.2, 4.3 and 4.4. We also report on a more comprehensive assessment in Section 4.5 on a larger set of matrices using simple criteria that are the convergence delay and the number of eigenpairs eventually computed within a prescribed maximum authorized delay. For those experiments, we chose a matrix arising from thermo-acoustic instability calculation in combustion chambers to illustrate the typical convergence behavior observed. Although its size is rather small ($n = 1.500$), it exhibits numerical difficulties that are encountered in real life large-scale problems [31, 32]; we refer to it as `thermo-acous` in the sequel. We also consider various matrices from the University of Florida (UF) test suite [11].

In order to study the numerical behavior of the IR strategies, and illustrate their possible robustness and weaknesses, we consider three additional executions in our numerical experiments. The first one is the non-faulty (NF) reference execution case. To distinguish between the interpolation quality effect and possible convergence delay introduced by the restart, we also report on what is referred to as the Enforced Restart (ER) execution. It consists in enforcing the solver to restart at iteration f using the available computed quantities at this iterations, that are:

- for the subspace iteration, ER is equivalent to NF because this numerical scheme is a fixed point iteration,
- for regular Arnoldi, it corresponds to classical Arnoldi with variable restarts,
- for IRAM and Jacobi-Davidson, the ER calculation is identical to the IR strategies except that none entries are interpolated; everything is computed using the values available at iteration f.

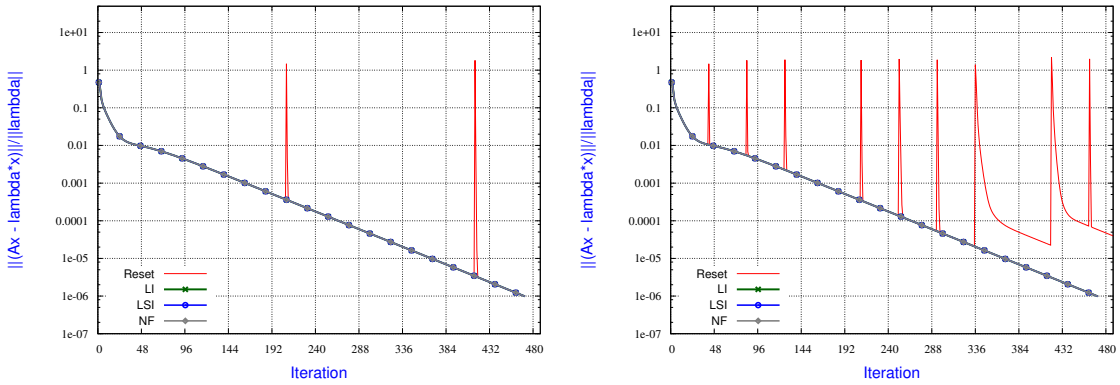
Finally, to illustrate the benefit and robustness of the interpolation policies, we report on the numerical behavior of a so-called Reset variant, where all the quantities interpolated by the IR strategies are instead substituted with random values.

The faulty parallel environment discussed in Section 2.1 is simulated with the following procedure as a sequence of crash nodes occurring at certain dates (iteration numbers). The iterations at which faults occur are decided following a pseudo-random Weibull probability distribution [40], considered as a relevant and realistic probabilistic model for characterizing the behavior of large-scale computational platforms [29]. We report results on the particular case of exponential distributions (corresponding to memory-less Weibull) with various mean time between failures (MTBF). At each fault, the identity of the crashed node is decided following a pseudo-random uniform distribution. We erase a proportion of dynamic data following a block row pattern, consistently with Assumption 1. By varying the proportion of lost data, we simulate a varying number of nodes; for instance, 1.0% of lost data virtually corresponds to 1 crashed node out of 100 running nodes. For the sake of fair comparison, the fault distributions are kept consistent between IR strategies, faults are injected at the same iterations (*e.g*, 210 and 416 in Figure 4.1) and during the same instructions for all resilient strategies as well as for ER and Reset (except NF, of course, for which no fault is injected).

4.1. Resilient subspace iteration methods to compute a few eigenpairs. In this section, we illustrate the robustness of the proposed resilient IR subspace iteration methods in the presence of faults. To investigate the robustness of our strategies, we simulate stressful conditions by increasing the fault rate and the volume of lost data. We present results for the subspace iteration method without polynomial acceleration (and refer to [2] for results when Chebyshev

acceleration is considered). We consider the calculation of the five eigenpairs corresponding to the largest magnitude eigenvalues (Figures 4.1 and 4.2). We report the maximum of the individual scaled residual norms (defined by Equation (4.1)) of the five Ritz pairs at each iteration. The execution ends when the five Ritz pairs satisfy the stopping criterion, i.e., when the maximum of the scaled residual norms is lower than the selected threshold ε; that is $\varepsilon = 10^{-6}$ for these experiments.

For computing the largest eigenvalues, NF converges in 470 iterations as depicted in Figures 4.1 and 4.2. The Reset strategy exhibits large peaks in the scaled residual norm after each fault, but succeeds to converge when only a few faults occur (Figure 4.1a) or only a small amount of data is lost (Figure 4.2a). Regarding the robustness of both IR strategies, the convergence histories of LI and LSI almost consistently overlap the NF curve regardless the fault rate (Figure 4.1) or the volume of lost data (Figure 4.2).



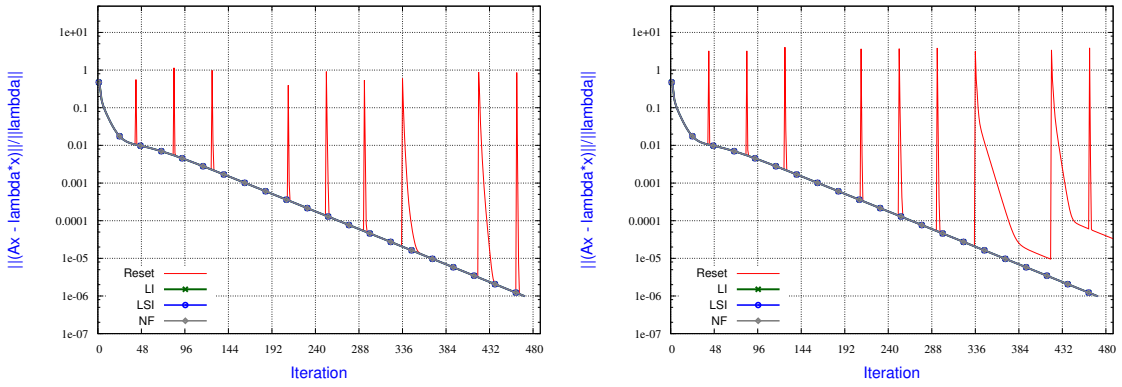
(a) 2 faults before the IR converge

(b) 9 faults before the IR converge

Fig. 4.1: **Impact of the fault rate** on the resilience of IR strategies when computing the **five eigenpairs** associated with the **largest eigenvalues** of `thermo-acous` using the **subspace iteration method**. A proportion of 0.8 % of data is lost at each fault. LI, LSI and NF convergence histories overlap.

4.2. Arnoldi method to compute one eigenpair. In this section, we assess the robustness of our resilient Arnoldi for computing the eigenpair associated with the smallest eigenvalue in magnitude of the UF matrix bcstk09. Because we look for the smallest eigenvalue, we consider a rather large restart parameter (m=50). We remind that one iteration accounts for building a Krylov basis of size m followed by the approximation of the desired eigenpair. We report the convergence histories in Figure 4.3. NF converges smoothly in 23 iterations whereas Reset strategy exhibits large peaks in the scaled residual norm after each fault. These peaks cannot be reduced significantly before the next fault occurs; consequently, Reset does not converge. Because ER has to restart more often than NF, its convergence history exhibits some delay compared to NF. On the other hand, both IR strategies are again extremely robust. Indeed, LI and LSI convergence coincide with ER. The convergence penalty is negligible when only 4 faults occur (one extra iteration) and moderate when 10 faults appear. The fact that LI and LSI convergence coincide with ER indicates that the spectral information regenerated by the LI and LSI is as good as the one computed by the regular solver. Although not reported in the paper, to keep it reasonably short, we can indicate that similar behaviors were observed for a larger amount of lost data (i.e., up to 8%).

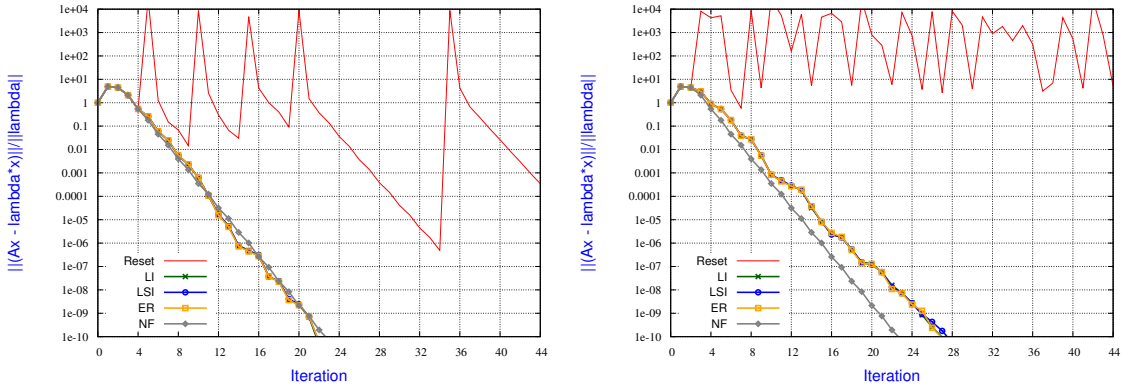
4.3. Implicitly restarted Arnoldi method to compute a few eigenpairs. To illustrate the robustness of IR strategies designed for IRAM, we compute the six eigenpairs ($nev = 6$) that



(a) 0.2% lost data

(b) 3% lost data

Fig. 4.2: **Impact of the amount of lost data** on the resilience of IR strategies when computing the **five eigenpairs** associated with the **largest eigenvalues** of `thermo-acous` using **subspace iteration method**. The volume of lost data varies from 0.2% to 3% whereas the fault rate is constant (9 faults). LI, LSI and NF coincide.



(a) 4 faults before the IR converge

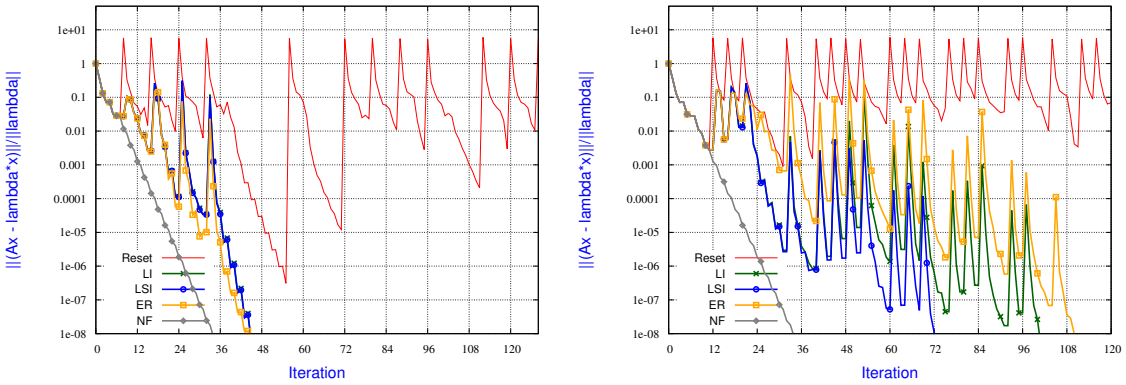
(b) 10 faults before the IR converge

Fig. 4.3: **Impact of the amount of lost data** on the resilience of IR strategies when converging the eigenpair associated with the **smallest eigenvalue** of HB-bcsstk09 using **Arnoldi method**. A proportion of 0.3 % of data is lost at each fault. LI, LSI and NF coincide in (a) and (b).

correspond to the largest magnitude eigenvalues of the matrix HB-dwr-2680. At each iteration, we report the maximum of the scaled residual norms of those six sought eigenpairs. We consider a restart parameter m=12 (see Algorithm 2). One iteration thus consists of building a Krylov subspace of size 12, followed by the computation of the approximate eigenpairs. If the eigenpairs do not satisfy the stopping criterion, the next iteration starts with a contracted Arnoldi equality of size $\tilde{m} = 6$.

The NF calculation computes the six sought eigenvectors in 35 iterations (index k in Algorithm 2), see Figure 4.4. As for regular Arnoldi, the Reset strategy exhibits a large peak in the scaled residual norm after each fault, its scaled residual norm increases further than the initial

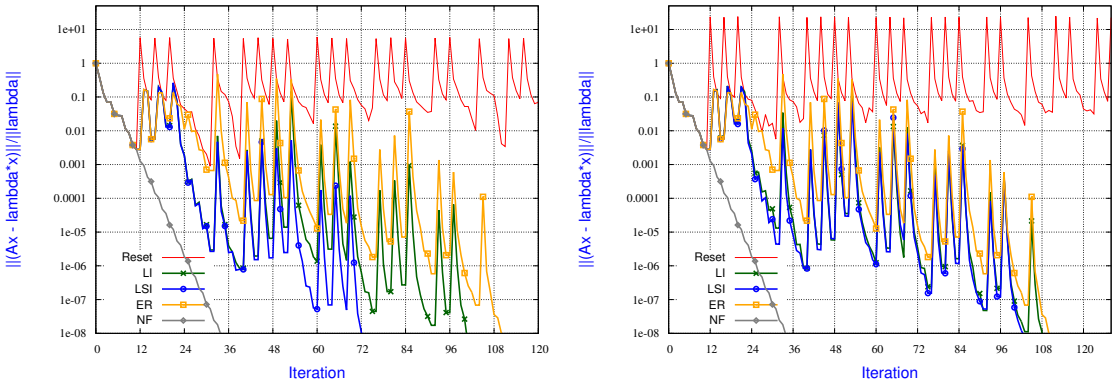
one. As a consequence, they prevent the convergence to take place even for a moderate MTBF (Figure 4.4a). On the other hand, both IR strategies are much more robust than Reset. However, they still require a few more iterations than NF. For a moderate MTBF (see Figure 4.4a) their convergence history is close to the one of ER, it can be concluded that this slight penalty is not due to the quality of interpolation but to the necessity of restarting with the information of the six dimension space compressed in one single direction. Because the convergence of IRAM depends on the spectral information included in the initial starting vector, one can observe that LI and LSI might even converge faster than ER (see Figure 4.4b). These convergence histories also confirm



(a) 4 faults before the IR converge

(b) 17 faults before the IR converge

Fig. 4.4: **Impact of the MTBF** on the resilience of IR strategies when computing the **six eigenpairs** with **largest eigenvalues** of HB-dwt-2680 using **IRAM**. MTBF varies whereas the amount of lost data per fault is constant (0.3 %).



(a) 0.3% of lost data

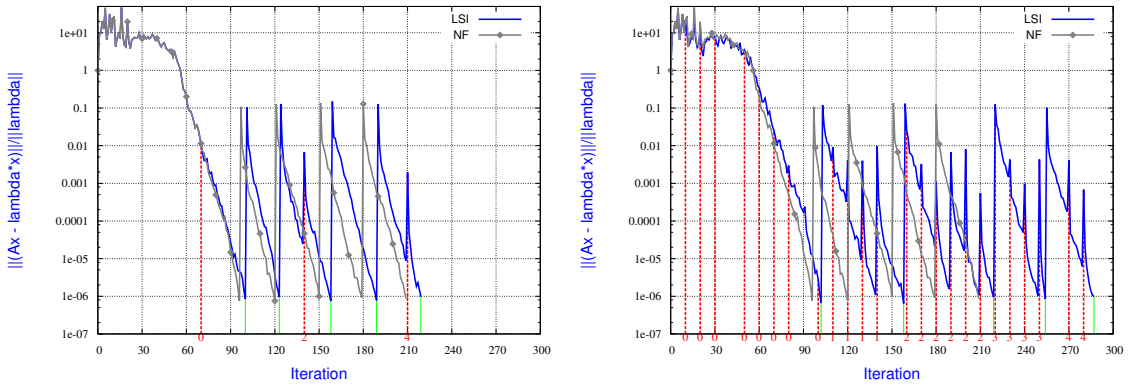
(b) 3% of lost data

Fig. 4.5: **Impact of the amount of lost data** on the resilience of IR strategies when computing the **six eigenpairs** with **largest eigenvalues** of HB-dwt-2680 using **IRAM**. The volume of lost data varies from 0.3% to 3% whereas the meant time between failure is constant (17 faults).

that the convergence delay grows with MTBF and the amount of lost data.

4.4. Jacobi-Davidson method to compute a few eigenpairs. In this section, we investigate the resilience of the IR strategies designed for Jacobi-Davidson. In these experiments, we seek the five ($nev = 5$) eigenpairs whose eigenvalues are the closest to zero ($\tau = 0$) for `thermo-acous`. The correction equation is solved using 30 iterations of GMRES. To facilitate the readability and the analysis of the convergence histories plotted in this section, we only report on results for one IR strategy and NF. In order to limit the number of graphs, we only consider LSI and refer to Section 4.5 for a more quantitative comparison between the different IR policies. In these plots, we use vertical green lines to indicate the convergence of new eigenpairs (such as iterations 100, 122, 187 and 218 in Figure 4.6a) and vertical red lines to indicate faulty iterations (such as iterations 70, 140 and 210 still in Figure 4.6a). According to Remark 1, although very likely to happen, there is no guarantee of retrieving all the already converged Schur vectors in the basis used to restart. As a consequence, we indicate the number of Schur vectors retrieved in the basis used to restart in red color under the vertical red line corresponding to the associated fault. For instance, 2 already converged Schur vectors are immediately retrieved at the restart, after the fault at iteration 140 in Figure 4.6a. In the Jacobi-Davidson method there is some flexibility in selecting the number of vectors (*i.e.*, the dimension of the space generated for restarting) that can be interpolated after a fault. For our experiments, we choose to interpolate the converged Schur vectors, denoted $nconv$, as well as nev of the best candidates for Schur vectors extracted from the search space V_f. For this example, the NF algorithm converges in 210 iterations while faulty executions have extra iterations.

In Figure 4.6 we depict the convergence histories when the fault rate varies leading to a number of faults that goes from 3 to 24; as expected, the larger the number of faults, the larger the convergence delay. However, the IR policy is rather robust and succeeds in converging the five eigenpairs in both cases. The influence of the volume of lost data is displayed in Figure 4.7 where



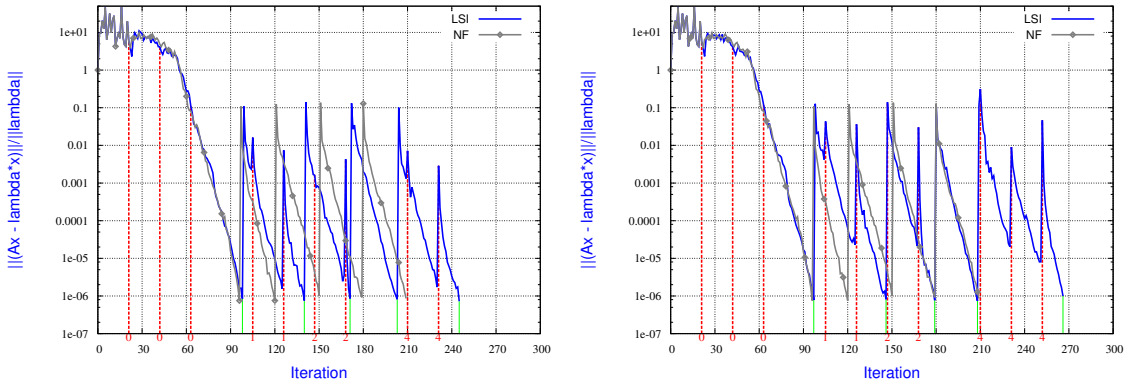
(a) 3 faults

(b) 24 faults

Fig. 4.6: **Impact of the fault rate** on the resilience of **LSI** using $(nev + nconv)$ regenerated vectors when computing the **five eigenpairs** associated with the **smallest eigenvalues** of `thermo-acous` using **Jacobi-Davidson**. The fault rate varies whereas a proportion of 0.2 % of data is lost at each fault. At each fault, all the already converged Schur vectors are retrieved in the basis of restart.

its proportion varies from 0.2 % to 3 % while the fault rate remains constant. As one would expect, the general trend is: the larger the amount of lost data, the larger the convergence penalty.

Despite their robustness, the proposed resilient schemes often induce peaks of the residual norm associated with the current best Schur Schur vector after each fault. This effect can be alleviated with a hybrid IR approach combined with checkpointing as follows. We interpolate the

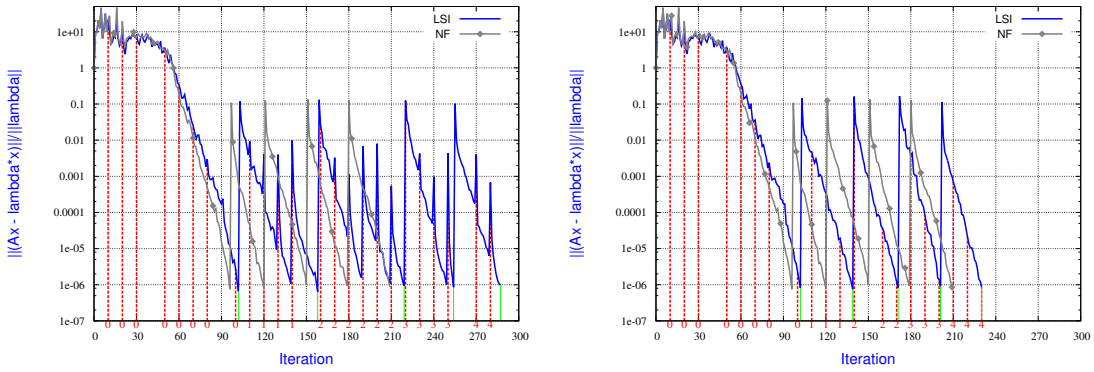


(a) 0.2% of lost data (9 faults)

(b) 3% of lost data (10 faults)

Fig. 4.7: **Impact of the amount of lost data** on the resilience of **LSI** using $(nev + nconv)$ regenerated vectors when computing the **five eigenpairs** associated with the **smallest eigenvalues** of `thermo-acous` using **Jacobi-Davidson**. The volume of lost data varies from 0.2% to 3% whereas the fault rate is constant. All converged Schur vectors are found immediately after interpolation followed by restart.

$nconv$ Schur vectors while reusing the best candidate Schur vector available in V_f (assuming we had checkpointed this single direction) when the fault occurs and interpolate other $nev - 1$ directions to recover the remaining meaningful spectral information from V_f. Peaks observed on the scaled residual norm with the baseline IR strategy (Figure 4.8a) are now alleviated with such a hybrid approach (Figure 4.8a).



(a) Baseline IR (24 faults)

(b) Hybrid IR (20 faults)

Fig. 4.8: **Impact of checkpointing the best Schur vector candidate** in the search space. Calculation of the **five eigenpairs** associated with the **smallest eigenvalues** using **Jacobi-Davidson** for `thermo-acous` with a proportion of 0.2 % of data is lost at each fault.

4.5. More comprehensive assessment. We now present a more comprehensive evaluation of the proposed IR strategies on a selection of matrices from UF collection for the eigensolvers that compute a few eigenvalues (i.e., all but Arnoldi). For each numerical method (and independently

from each other), we selected a set of matrices (and an associated objective nev number of eigenpairs to compute) from that collection that had a "reasonable" convergence behavior in the NF case. We then assess the IR methods on these matrices with various MTBF and proportions of data loss ; we report the delay on convergence in terms of a number of iterations normalized by the NF reference. We authorize faulty executions to run until a maximum defined as $\texttt{max} - \texttt{delay}$ in the sequel whose value is set specifically for each eigensolver. If the IR method failed to recover all the wanted nev eigenpairs after this authorized delay we report the number of possibly converged eigenpairs.

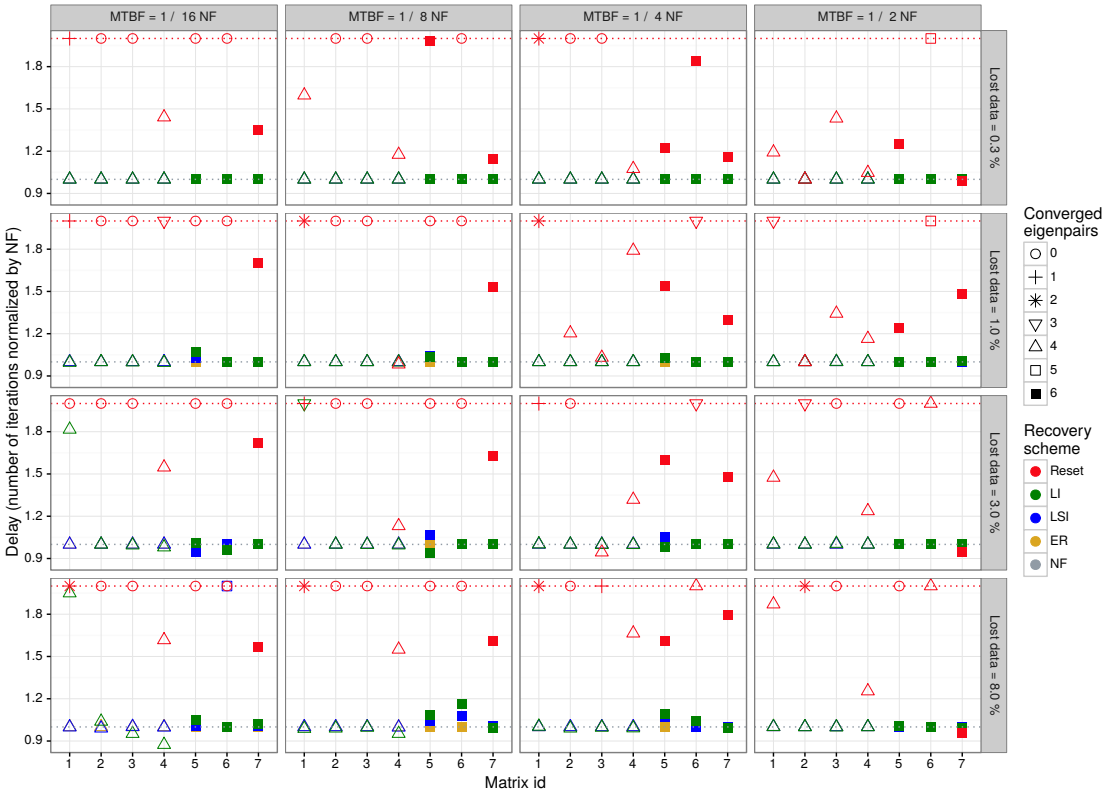


Fig. 4.9: Delay of IR **subspace iteration** policies over NF with an authorized delay $\texttt{max} - \texttt{delay} = 2.0$. Considered matrices: 1=HB-bcsstk09 - 2=HB-dwt-2680 - 3=Bai-rdb5000 - 4=DRIVCAV-cavity01 - 5=DRIVCAV-cavity21 - 6=Sandia-fpga-dcop-01 - 7=CPM-cz628. For matrices $1-4$ we aim at converging $nev = 4$ eigenpairs while for matrices $5-7$ we aim at converging $nev = 6$ eigenpairs.

For instance, for the subspace iteration method to compute the largest eigenvalues, we selected seven matrices and report the delay on their convergence in the presence of faults in Figure 4.9. For the first four matrices, we aim at converging $nev = 4$ eigenpairs while for the last three matrices we aim at converging $nev = 6$ eigenpairs. A maximum delay $\texttt{max} - \texttt{delay} = 2.0$ (dotted red line) is authorized. The MTBF is set up as a proportion of the number of iterations required by NF, varying from MTBF = 1/16 NF (highest fault rate and leftmost column in Figure 4.9) to MTBF = 1/2 NF (lowest fault rate, rightmost column). The proportion of lost data varies from 0.3% (topmost row) to 8.0% (bottom-most row). With a relatively low fault rate (MTBF = 1/2 NF) and proportion of lost data (0.3%), the top right corner of Figure 4.9 shows that the impact on the delay of IR methods is consistently negligible. Indeed, both LI (green data points) and LSI

(blue data points, here overlapped with green data points) converge all the $nev = 4$ (triangles) sought eigenpairs for the first four matrices and $nev = 6$ (plain squares) wanted eigenpairs for the last three matrices with a relative delay of approximately 1.0 (meaning no delay over NF). The delay of Reset (red data points) is negligible for matrix 2 and 7, moderated for matrices 1, 3, 4 and 5 while matrix 6 converges only 5 (empty square) out of 6 (plain squares) once the authorized delay ($\texttt{max} - \texttt{delay} = 2.0$) has been reached. The proposed IR strategies remain extremely robust when the fault rate or the amount of lost data are increased as the delay remains negligible in most cases and never exceeds 1.2 in any case. Reset is much more sensitive to those variations but manage to converge all wanted eigenpairs for two matrices (matrices 4 and 7) out of seven before the authorized delay even with a very high fault rate (MTBF = 1/16 NF) for all considered proportions of lost data (leftmost column). These results confirm the natural resilience of the subspace iteration method, as fixed point scheme, qualitatively observed in Section 4.1.

Figure 4.10 shows the delay of the proposed IRAM resilient IR strategies using a similar

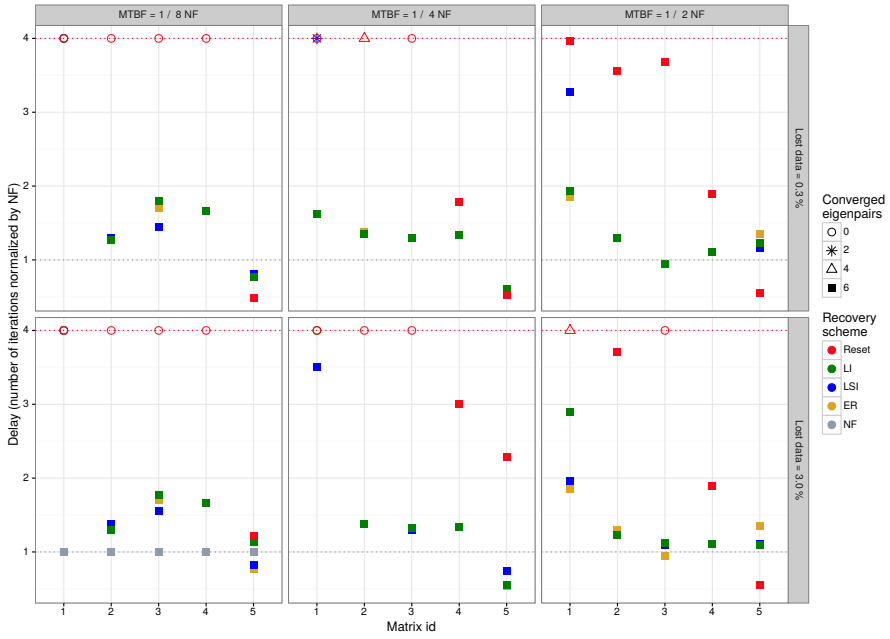


Fig. 4.10: Delay of IR **IRAM** policies over NF with an authorized delay $\texttt{max} - \texttt{delay} = 4.0$. Considered matrices: 1=DRIVCAV-cavity01 - 2=DRIVCAV-cavity21 - 3=HB-dwt-2680 - 4=Rajat-rajat04 - 5=Nasa-nasa4704. For all matrices, we aim at converging $nev = 6$ eigenpairs.

experimental framework when we attempt to compute nev largest eigenvalues. For all matrices we aim at converging $nev = 6$ eigenpairs within an authorized delay $\texttt{max} - \texttt{delay} = 4.0$. The NF behavior is reported in the bottom left plot as a reference: matrices converge all $nev = 6$ (plain square, light gray data points) in a 1.0 delay when no fault occurs (by definition). The top right corner of Figure 4.10 shows that the delay of IRAM resilient strategies is moderated for matrices $2-5$. Indeed, both LI (green data points) and LSI (blue data points, again overlapped or partially overlapped with green data points) converge all the $nev = 6$ (plain squares) wanted eigenpairs with a relative delay over NF below 1.2. Matrix 1 induces more significant delays (1.9 and 3.2, respectively for LI and LSI). When the fault rate is increased (MTBF = 1/4 NF or 1/8 NF, second and first columns, respectively), the delay of IR strategies is more significant but remains strictly below than 2.0 for matrices $2-5$. Matrix 1 is again more difficult to converge with those fault rates. With an intermediate fault rate (MTBF = 1/4 NF), LI succeeds to converge all eigenpairs in

one case (data loss = 0.3%) but fails to converge any eigenpair in the other case (data loss = 3%). On the contrary, with the same fault rate, LSI succeeds to converge (late, 3.5 delay, but within the authorized delay) all eigenpairs in that latter case (data loss = 3%, MTBF=1/4 NF) and only converges 2 eigenpairs out of 6 in the former case (data loss = 0.3%, MTBF = 1/4 NF). With an even higher fault rate (MTBF = 1 / 8 NF), LI and LSI consistently fail to converge any eigenpair for Matrix 1. The fact that even ER does not converge any eigenpair in that situation (blue, green, red and yellow circles overlap, meaning that none of the LSI, LI, Reset nor ER associated strategies has converged a single eigenpair) shows that the limitation is not due to the interpolation scheme but to the necessity of restarting. Furthermore, the low robustness of Reset (with the notable exception of Matrix 5 whose convergence can surprisingly be enhanced with both IR schemes as well as Reset) even for configurations where LI and LSI achieve an excellent convergence confirm the relevance of the proposed interpolations schemes. Figure 4.10 shows that the impact on the delay of IR methods is moderated for matrices $2-5$. Indeed, both LI (green data points) and LSI (blue data points, again overlapped or partially overlapped with green data points) converge all the $nev = 6$ (plain squares) wanted eigenpairs with a relative delay over NF below 1.2. Matrix 1 induces more significant delays (1.9 and 3.2, respectively for LI and LSI). To conclude on the experiments with IRAM, we have also performed tests with a somehow simpler approach that consists in interpolating the nev Ritz pairs without prior application of the shifted QR filtering. To keep the paper short, we did not report any results but indicate that this latter approach is slightly less robust in particular when there is a large number of faults.

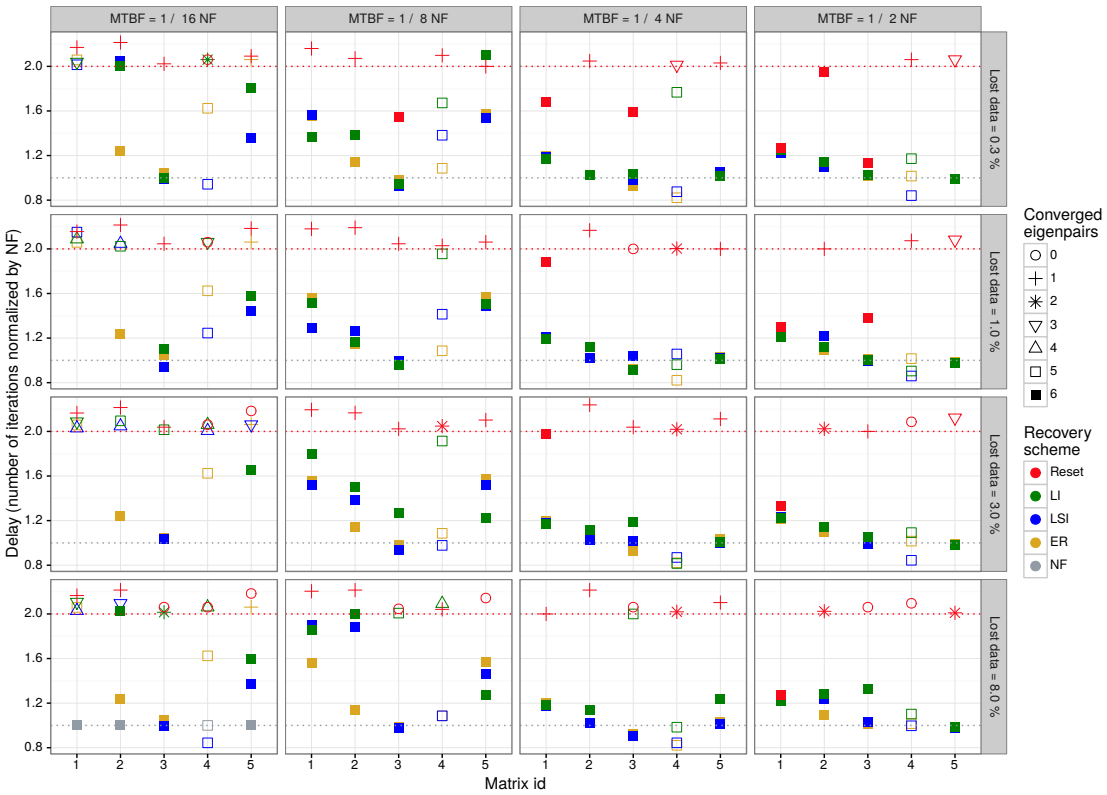


Fig. 4.11: Delay of **Jacobi-Davidson** IR policies over NF with an authorized delay `max - delay` = 2.0. Considered matrices: 1=Bai-qc324 - 2=CPM-cz628 - 3=DRIVCAV-cavity01 - 4=FIDAP-ex22 - 5=HB-bcsstk09. For matrices $1-3$ and 5 we aim at converging $nev = 6$ eigenpairs while for matrix 4 we aim at converging $nev = 5$ eigenpairs.

In the case of Jacobi-Davidson, it may happen that a fault just occurred while approaching $\texttt{max} - \texttt{delay}$. In that case, the IR method is authorized to perform a few additional iterations in order to retrieve already converged eigenpairs and we include those extra iterations in the reported delay. Figure 4.11 shows the delay of the proposed resilient Jacobi-Davidson IR strategies on another set of matrices. For matrices $1-3$ and 5 we aim at converging $nev = 6$ eigenpairs (associated with eigenvalues closest to 0) while for Matrix 4 we aim at converging $nev = 5$ eigenpairs, as represented (only in the bottom left corner) by the NF convergence status with plain gray squares for matrices $1-3$ and 5 and an empty gray square for Matrix 4. Because the resilient Jacobi-Davidson methods appeared to be more robust than IRAM in Section 4.4, we performed experiments with a range of MTBF and data loss as wide as in the case of the extremely resilient subspace iteration scheme with the same value $\texttt{max} - \texttt{delay} = 2.0$ of authorized delay. These more comprehensive results confirm the robustness of the proposed Jacobi-Davidson resilient schemes. They also confirm the higher sensitivity to the quality of the interpolation. In particular, the LSI-based Jacobi-Davidson resilient solver achieves a better convergence than its LI-based alter ego. Indeed, it converges all sought eigenpairs with a very limited delay (below than 1.2) for all considered matrices when the fault rate is low (MTBF = 1/2 NF, right-most column) or intermediate (MTBF = 1/4 NF, third column) . With an higher fault rate (MTBF = 1/8 NF, second column), this solver still succeeds to converge all eigenpairs of all matrices within the authorized delay $\texttt{max} - \texttt{delay} = 2.0$. On the other hand, LI is more sensitive to the proportion of lost data; for instance it misses one eigenpair in the authorized delay for a similar fault rate (MTBF = 1/8) and a high data loss rate (8.0%) for matrices 3 (empty green square) and 4 (green triangle). The importance of the quality of the interpolation is also highlighted with the superiority of ER over IR strategies for large proportions of fault rate (last row). The Reset strategy could compute all wanted eigenpairs of three matrices out of five with minimum fault rate and lost data (MTBF = 1/2 NF and 0.3% of lost data, top right) but is extremely sensitive to both parameters; for instance, it fails to converge more than one eigenpair with MTBF = 1/8 NF and 1.0% of data loss for any matrix.

The overall robustness of the Jacobi-Davidson resilient solver can be further improved using the hybrid technique that combines IR with checkpointing the best candidate at each iteration. The corresponding results are reported in Figure 4.12. For instance, in the worst case (NF = 1/16 NF and 8.0% of lost data, bottom left), the LSI-based resilient hybrid Jacobi-Davidson solver succeeds to converge in time all eigenpairs for four matrices out of five. Note that the quality of the interpolation remains very important. LI converges all eigenpairs for only one matrix (Matrix 5) in the same configuration.

5. Concluding remarks. Many scientific and engineering applications require the computation of eigenpairs of large sparse matrices. The objective of the paper has been to propose and study numerical schemes suitable for the design of resilient parallel eigensolvers. For that purpose, we have proposed two interpolation procedures to regenerate meaningful spectral information for restarting the eigensolver after a fault. To evaluate the qualitative behavior of the resilient schemes, we have simulated stressful conditions by increasing the fault rate and the volume of data loss.

We have considered the subspace iteration method for the computation of eigenpairs corresponding to the largest magnitude eigenvalues. For this method, the Reset strategy often strongly penalizes the convergence at each fault, while both LI and LSI are extremely robust and resilient, regardless of the number of faults and the volume of data loss. The same numerical behavior is observed for our resilient Arnoldi. Our LI/LSI resilient IRAM for the computation of a few eigenpairs are much more robust than Reset. However, they do exhibit a slight penalty, not due to the quality of interpolation, but to the restarting policy that leads to compressing in a single direction the eigenspace under calculation when faults occur.

We have had a stronger emphasis on the Jacobi-Davidson method. The motivation is twofold: the Jacobi-Davidson method is widely used in many real-life applications, and, in addition, it offers some flexibility to select different spectral information to construct an efficient restart mechanism

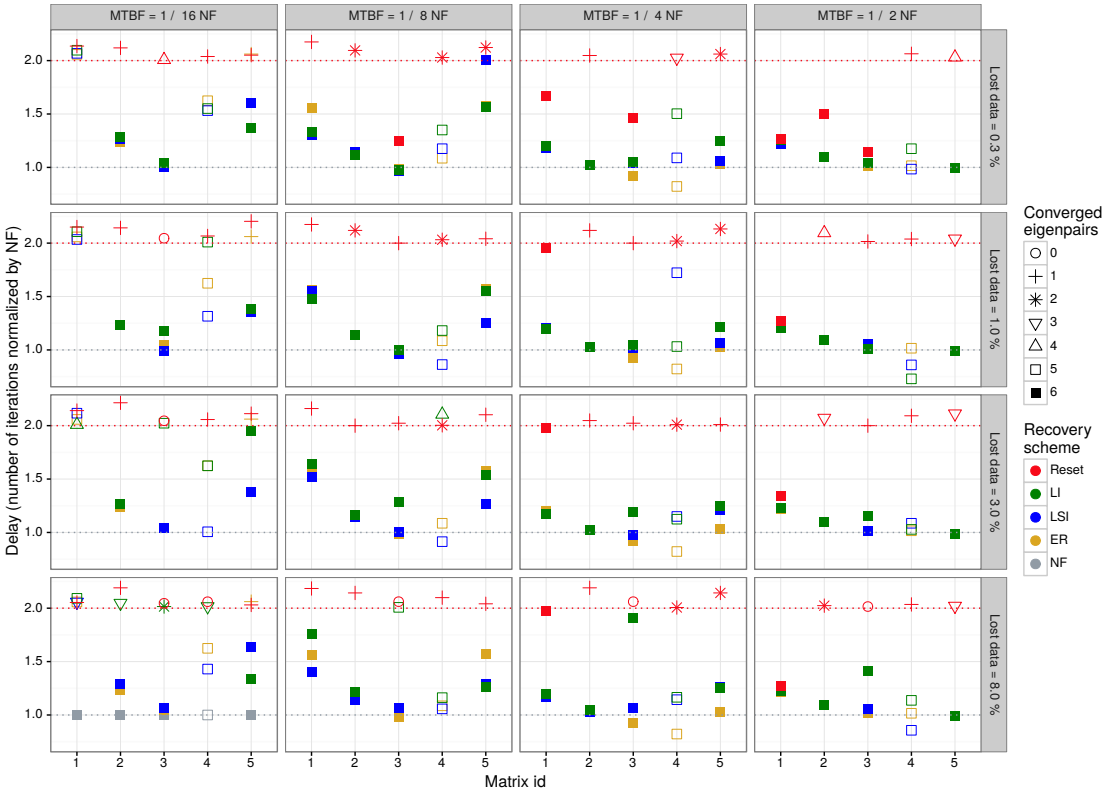


Fig. 4.12: Delay of the **hybrid Jacobi-Davidson** IR policies combined with checkpointing of the best Schur candidate over NF with an authorized delay $\texttt{max} - \texttt{delay} = 2.0$. Considered matrices: 1=Bai-qc324 - 2=CPM-cz628 - 3=DRIVCAV-cavity01 - 4=FIDAP-ex22 - 5=HB-bcsstk09. For matrices $1-3$ and 5 we aim at converging $nev = 6$ eigenpairs while for matrix 4 we aim at converging $nev = 5$ eigenpairs.

after a fault. We have observed that despite the increase of the amount of recovered data, the peak of the residual norm associated with the current Schur vector persists after a fault. For a possible remedy of these effects, we have designed a hybrid approach that consists in combining interpolation techniques with classical checkpoint for a single vector. This illustrates that numerical resilient strategies can be effectively combined with the state-of-the-art fault tolerant policies to design efficient and robust methods as demonstrated for the Jacobi-Davidson algorithm by the combination of light checkpointing and a numerical resilience.

A deeper analysis, which will be carried out in a future work, will be to analyze the robustness of the IR methods with respect to the numerical properties of the targeted eigenpairs such as the clustering or conditioning. We nevertheless believe that establishing a general theoretical framework to study the IR strategies is challenging and was clearly out of the scope of this work, which was aimed at illustrating possible remedies to introduce some numerical resiliency in some of the most widely used eigensolvers.

Finally, the effectiveness of the proposed approaches has been illustrated from a numerical viewpoint. Their effectiveness and scalable implementation in a parallel computing environment deserve to be studied once the fault tolerance supports is included in the MPI standard.

Acknowledgments. We would like to thank the anonymous referees whose constructive comments enabled us to improve the readability of this paper. Finally, this work was partially supported

by the French research agency ANR in the framework of the RESCUE project (ANR-10-BLANC-0301), in particular the PhD thesis of the fourth author was funded by this project. This research also benefited from the G8-ECS project.

REFERENCES

[1] Emmanuel Agullo, Luc Giraud, Abdou Guermouche, Jean Roman, and Mawussi Zounon. Towards resilient parallel linear Krylov solvers: recover-restart strategies. Research Report RR-8324, Inria, July 2013. Preliminary version of a paper to appear in Numerical Linear Algebra with Applications.

[2] Emmanuel Agullo, Luc Giraud, Pablo Salas, and Mawussi Zounon. On resiliency in some parallel eigensolvers. Research Report 8625, Inria, 2015.

[3] Emmanuel Agullo, Luc Giraud, and Mawussi Zounon. On the resilience of parallel sparse hybrid solvers. In *22nd IEEE International Conference on High Performance Computing, HiPC 2015, Bengaluru, India, December 16-19, 2015*, pages 75–84. 2015. doi:10.1109/HiPC.2015.9.

[4] Lorenzo Alvisi and Keith Marzullo. Message logging: Pessimistic, optimistic, causal, and optimal. *IEEE Trans. Softw. Eng.*, 24(2):149–159, February 1998. ISSN 0098-5589. doi:10.1109/32.666828.

[5] J. Anfinson and F. T. Luk. A linear algebraic model of algorithm-based fault tolerance. *IEEE Trans. Comput.*, 37:1599–1604, December 1988. ISSN 0018-9340. doi:10.1109/12.9736.

[6] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Q. Appl. Math*, 9(17):17–29, 1951.

[7] Todd M. Austin. DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design. In *Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture*, MICRO 32, pages 196–207. IEEE Computer Society, Washington, DC, USA, 1999. ISBN 0-7695-0437-X.

[8] Anita Borg, Jim Baumbach, and Sam Glazer. A message system supporting fault tolerance. *SIGOPS Oper. Syst. Rev.*, 17(5):90–99, October 1983. ISSN 0163-5980. doi:10.1145/773379.806617.

[9] Franck Cappello, Henri Casanova, and Yves Robert. Preventive migration vs. preventive checkpointing for extreme scale supercomputers. *Parallel Processing Letters*, pages 111–132, 2011.

[10] Zizhong Chen. Online-ABFT: an online algorithm based fault tolerance scheme for soft error detection in iterative methods. In *ACM SIGPLAN Notices*, volume 48, pages 167–176. ACM, 2013.

[11] Timothy A. Davis and Yifan Hu. The University of Florida sparse matrix collection. *j-TOMS*, 38(1):1:1–1:25, November 2011.

[12] E. N. (Mootaz) Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and David B. Johnson. A Survey of Rollback-recovery Protocols in Message-passing Systems. *ACM Comput. Surv.*, 34(3):375–408, September 2002. ISSN 0360-0300. doi:10.1145/568522.568525.

[13] E.N. Elnozahy, D.B. Johnson, and W. Zwaenepoel. The performance of consistent checkpointing. In *11th Symposium on Reliable Distributed Systems, 1992. Proceedings.*, pages 39–47. Oct 1992. doi: 10.1109/RELDIS.1992.235144.

[14] Diederik R. Fokkema, Gerard L. G. Sleijpen, and Henk A. Van der Vorst. Jacobi-Davidson style QR and QZ algorithms for the partial reduction of matrix pencils. *SIAM J. SCI. COMPUT*, 20:94–125, 1996.

[15] John A. Gunnels, Robert A. Van De Geijn, Daniel S. Katz, and Enrique S. Quintana-ortí. Fault-tolerant high-performance matrix multiplication: Theory and practice. In *Dependable Systems and Networks*, pages 47–56. 2001.

[16] G.W. Stewart. *Matrix algorithms – Volume II: Eigensystems*. SIAM, 2001.

[17] Kuang-Hua Huang and J. A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Trans. Comput.*, 33:518–528, June 1984. ISSN 0018-9340.

[18] R.K. Iyer, N.M. Nakka, Z.T. Kalbarczyk, and S Mitra. Recent advances and new avenues in hardware-level reliability support. *Micro, IEEE*, 25(6):18–29, Nov 2005. ISSN 0272-1732. doi:10.1109/MM.2005.119.

[19] L. Jaulmes, M. Casas, M. Moretó, E. Ayguade, J. Labarta, and M. Valero. Exploiting asynchrony from exact forward recovery for DUE in iterative solvers. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15)*. Austin, USA, November 2015.

[20] David B Johnson and Willy Zwaenepoel. *Sender-based message logging*. 1987.

[21] Julien Langou, Zizhong Chen, George Bosilca, and Jack Dongarra. Recovery Patterns for Iterative Methods in a Parallel Unstable Environment. *SIAM J. Sci. Comput.*, 30:102–116, November 2007. ISSN 1064-8275. doi:10.1137/040620394.

[22] R.B. Lehoucq and D. C. Sorensen. Deflation techniques for an implicitly restarted Arnoldi iteration. *SIAM J. Matrix Anal. Appl*, 17:789–821, 1996.

[23] C.-C.J. Li and W.K. Fuchs. Catch-compiler-assisted techniques for checkpointing. In *Fault-Tolerant Computing, 1990. FTCS-20. Digest of Papers., 20th International Symposium*, pages 74–81. June 1990. doi: 10.1109/FTCS.1990.89337.

[24] Yudan Liu, R. Nassar, C.B. Leangsuksun, N. Naksinehaboon, M. Paun, and S.L. Scott. An optimal checkpoint/restart model for a large scale high performance computing system. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2008)*, pages 1–9. April 2008. ISSN 1530-2075. doi:10.1109/IPDPS.2008.4536279.

[25] N. Oh, P.P. Shirvani, and E.J. McCluskey. Error detection by duplicated instructions in super-scalar processors.

Reliability, IEEE Transactions on, 51(1):63–75, Mar 2002. ISSN 0018-9529. doi:10.1109/24.994913.

[26] J. S. Plank, Y. Kim, and J. Dongarra. Fault tolerant matrix operations for networks of Workstations Using Diskless Checkpointing. *Journal of Parallel and Distributed Computing*, 43(2):125–138, June 1997.

[27] James Plank. An overview of checkpointing in uniprocessor and distributed systems, focusing on implementation and performance. Technical Report UT-CS-97-372, Dept of Computer science, University of Tennessee, 1997.

[28] J.S. Plank and K. Li. ICKP: a consistent checkpointer for multicomputers. *Parallel Distributed Technology: Systems Applications, IEEE*, 2(2):62–67, Summer 1994. ISSN 1063-6552. doi:10.1109/88.311574.

[29] Narasimha Raju, Gottumukkala, Yudan Liu, Chokchai B. Leangsuksun, Raja Nassar, and Stephen Scott. Reliability Analysis in HPC clusters. *Proceedings of the High Availability and Performance Computing Workshop*, 2006.

[30] Yousef Saad. *Numerical Methods for Large Eigenvalue Problems.* Manchester University Press, Manchester, UK, 1992.

[31] Pablo Salas. *Physical and numerical aspects of thermoacoustic instabilities in annular combustion chambers.* Ph.D. thesis, Université Bordeaux 1, November 2013.

[32] Pablo Salas, Luc Giraud, Yousef Saad, and Stéphane Moreau. Spectral recycling strategies for the solution of nonlinear eigenproblems in thermoacoustics. *Numerical Linear Algebra with Applications*, 22(6):1039–1058, 2015. doi:10.1002/nla.1995.

[33] J.C. Sancho, F. Petrini, K. Davis, R. Gioiosa, and S. Jiang. Current practice and a direction forward in checkpoint/restart implementations for fault tolerance. In *Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium.* April 2005. doi:10.1109/IPDPS.2005.157.

[34] M. Scholzel. Reduced Triple Modular redundancy for built-in self-repair in VLIW-processors. In *Signal Processing Algorithms, Architectures, Arrangements and Applications, 2007*, pages 21–26. Sept 2007. doi:10.1109/SPA.2007.5903294.

[35] Gerard L. G. Sleijpen and Henk A. Van der. A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM Rev.*, 42(2):267–293, June 2000.

[36] T.N. Vijaykumar, I Pomeranz, and K. Cheng. Transient-fault recovery using simultaneous multithreading. In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pages 87–98. 2002. ISSN 1063-6897. doi:10.1109/ISCA.2002.1003565.

[37] John von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata Studies*, pages 43–98, 1956.

[38] Chao Wang, Frank Mueller, Christian Engelmann, and Stephen L. Scott. Hybrid full/incremental checkpoint/restart for MPI jobs in HPC environments. In *Dept. of Computer Science, North Carolina State University.* 2009.

[39] Chris Weaver and Todd M. Austin. A Fault Tolerant Approach to Microprocessor Design. In *Proceedings of the 2001 International Conference on Dependable Systems and Networks (Formerly: FTCS)*, DSN '01, pages 411–420. IEEE Computer Society, Washington, DC, USA, 2001. ISBN 0-7695-1101-5.

[40] Wallodi Weibull. A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, 18:293–297, 1951.

[41] Mawussi Zounon. *On numerical resilience in linear algebra.* Theses, Université de Bordeaux, April 2015.